Demos

Peng Chen and Enrique del Castillo

We provide some examples to demonstrate the following R functions.

Table 1. Main OptimaRegion functions related to this research.

Function	**Objective**
BayesOptRegionQuad	Compute a credible region on the global optimum of a quadratic polynomial model in 2 variables
BayesOptRegionGP	Compute a credible region on the global optimum of a Gaussian process model in 2 variables
OptRegionQuad	Compute a distribution-free confidence region on the global optimum of a quadratic polynomial model in 2 variables
GloptiPolyRegion	Compute a distribution-free confidence region on the global optimum of a polynomial model up to cubic order in $3 \sim 5$ variables
OptRegionTps	Compute a distribution-free confidence region on the global optimum of a thin-plate spline model in 2 variables
OptRegionGP	Compute a confidence region on the global optimum of a Gaussian process model in 2 variables
OptRegionOK	Compute a distribution-free confidence region on the global optimum of an ordinary Kriging model in 2 variables
OptRegionPC1Quad	Compute a distribution-free confidence region on the global optimum of the dominant hidden quadratic function of a system with high-dimensional responses
GloptipolyR	R implementation of the "Gloptipoly" algorithm for global optimization of polynomial functions subject to bounds

To run the examples, load packages:

```
library(mvtnorm)
library(lhs)
library(nloptr)
library(parallel)
library(microbenchmark)
library(grDevices)
library(keras)
library(geometry)
library(rjags)
library(fields)
library(gridExtra)
library(MASS)
library(rlang)
library(tidyverse)
library(progress)
```

```
library(DepthProc)
library(metR)
```

source files:

```
source("BayesOptRegionQuad.R")
source("BayesOptRegionGP.R")
source("OptRegionQuad.R")
source("GloptiPolyRegion.R")
source("OptRegionTps.R")
source("OptRegionGP.R")
source("OptRegionOK.R")
source("OptRegionPC1Quad.R")
source("GloptiPolyR.R")
```

and load datasets:

```
load("dataset_1.RData")
load("dataset_2.RData")
load("dataset_3.RData")
load("dataset_4.RData")
load("dataset_5.RData")
load("dataset_6.RData")
load("dataset_7.RData")
load("dataset_8.RData")
```

Function 1. `BayesOptRegionQuad`

Function `BayesOptRegionQuad` computes a credible region on the global optimum of a quadratic polynomial model in 2 variables To call it, use command:

```
res <- BayesOptRegionQuad(
  design = dataset_1$design, y = dataset_1$y,
  alpha = 0.05, n_post = 500,
  constr_lb = c(-sqrt(2), -sqrt(2)), constr_ub = c(sqrt(2), sqrt(2)),
  parallel = TRUE
)
```

where `design` takes the design matrix, `y` takes the observation vector, `constr_lb` and `constr_ub` take the lower and upper bounds of the experimental region, `alpha` specifies the acceptable Type-I error, `n_post` specifies the number of posterior parameter draws, and `parallel` specifies if it utilizes multiple cores to compute for the credible region. The function returns a list. Use command `str(res)` to check its structure:

```
List of 5
 $ optima   :'data.frame': 475 obs. of  2 variables:
  ..$ x1: num [1:475] -0.148 -0.172 -0.337 -0.319 -0.51 ...
  ..$ x2: num [1:475] -0.2289 -0.0685 -0.049 -0.0415 0.0879 ...
 $ opt_hat  :'data.frame': 1 obs. of  2 variables:
  ..$ x1: num -0.232
  ..$ x2: num -0.107
 $ beta_hat : Named num [1:6] 90.058 -1.232 -0.682 -2.44 -2.157 ...
  ..- attr(*, "names")= chr [1:6] "intercept" "x1" "x2" "x1x1" ...
 $ constr_lb: num [1:2] -1.41 -1.41
```

```
 $ constr_ub: num [1:2] 1.41 1.41
 - attr(*, "class")= chr "bayescrquad"
```

where `optima` contains all the simulated posterior optima, `opt_hat` contains the point estimate of the optimum, and `beta_hat` contains the point estimate of the polynomial coefficients. Use command:

```
plot(res, xlab = "x1", ylab = "x2")
```

to draw the credible region on the optimum, as shown in Figure 1. The contours

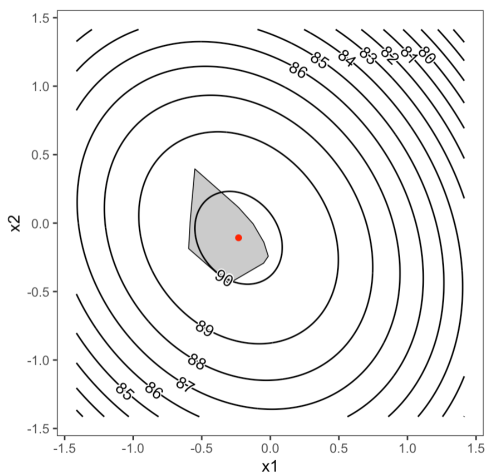

Figure 1. A credible region generated by `BayesOptRegionQuad`.

indicate the point estimate of the response surface. The red dot indicates the point estimate of the optimum. The gray convex hull indicates the credible region on the optimum.

Function 2. `BayesOptRegionGP`

Function `BayesOptRegionGP` computes a credible region on the global optimum of a Gaussian process model in 2 variables. To call it, use command:

```
c(thetas, thetas_thin, credible_region) %<-% BayesOptRegionGP(
  design = dataset_2$design, y = dataset_2$y,
  constr_lb = c(0, 0), constr_ub = c(5, 5), alpha = 0.05,
  chain_length = 1e5, thin_interval = 200, n_path_per_theta = 1,
  xi = 0.1, base_grid_size = 225,
  process_mean_lb = -20, process_mean_ub = 20,
  length_scale_lb = 0, length_scale_ub = 100,
  fun_var_lb = 0, fun_var_ub = 60, noise_var_lb = 0, noise_var_ub = 2,
  parallel = TRUE
)
```

where `chain_length` specifies the length of the stabilized Markov chain, `thin_interval` specifies how often to take a sample from the chain to thin it, `n_path_per_theta` specifies how many sample paths to simulate based on each posterior draw of the GP parameters, `xi` specifies the penalization parameter of a point-selection algorithm, and `process_mean_lb` ... `noise_var_ub` specify the lower and upper bounds for the prior distributions. The function returns three lists, `thetas`, `thetas_thin`, and `credible_region`. List `thetas` contains the information of the full Markov chain. Use command `str(thetas)` to check its structure:

```
List of 4
 $ process_mean: num [1:100000] 0.266 1.382 1.283 0.941 1.067 ...
 $ length_scale: num [1:100000] 81.2 95.3 67.8 98.7 65.8 ...
 $ fun_var     : num [1:100000] 1.893 0.929 1.061 0.959 0.874 ...
 $ noise_var   : num [1:100000] 0.0201 0.022 0.0226 0.0242 0.0233 ...
 - attr(*, "class")= chr "mcmcdraw"
 - attr(*, "row.names")= int [1:100000] 1 2 3 4 5 6 7 8 9 10 ...
```

use command `summary(thetas)` to check selected percentiles of the posterior distributions:

```
                   2.5%          25%          50%          75%       97.5%
process_mean -2.707894e+00 -0.8902202248 -0.089409026  0.689365899  2.51150974
length_scale  1.389972e+01 48.4875932325 69.904595842 86.265334856 98.71970585
fun_var       3.274057e-01  1.1418696835  1.688102322  2.256419015  3.53513809
noise_var     6.005339e-05  0.0007298583  0.001833649  0.003934859  0.01216852
```

and use command `plot(thetas)` to generated the trace and density plots of the the posterior distributions, as shown in Figure 2. List `thetas_thin` contains the information of the thinned Markov chain and can be explored in the same way. List `credible_region` contains the information of the credible region, use command `str(credible_region)` to check its structure:

```
List of 5
 $ optima   :'data.frame': 475 obs. of  2 variables:
  ..$ x1: num [1:475] 1.07 1.07 1.48 1.44 1.13 ...
  ..$ x2: num [1:475] 1.43 1.43 1.67 1.31 1.38 ...
 $ opt_hat  :'data.frame': 1 obs. of  2 variables:
  ..$ x1: num 1.66
  ..$ x2: num 1.41
 $ rs_hat   :function (x)
  ..- attr(*, "srcref")= 'srcref' int [1:8] 146 3 152 3 3 3 146 152
  .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile'
        <environment: 0x7fa6ac1d6f28>
 $ constr_lb: num [1:2] 0 0
 $ constr_ub: num [1:2] 5 5
 - attr(*, "class")= chr "bayescrgp"
```

where `rs_hat` contains the point estimate of the response surface. Use command:

```
plot(credible_region, xlab = "x1", ylab = "x2")
```

to draw the credible region on the optimum, as shown in Figure 3.

Function 3. `OptRegionQuad`

Function `OptRegionQuad` computes a distribution-free confidence region on the global optimum of a quadratic polynomial model in 2 variables. To call it, use command:

```
res <- OptRegionQuad(
  design = dataset_3$design, y = dataset_3$y,
```

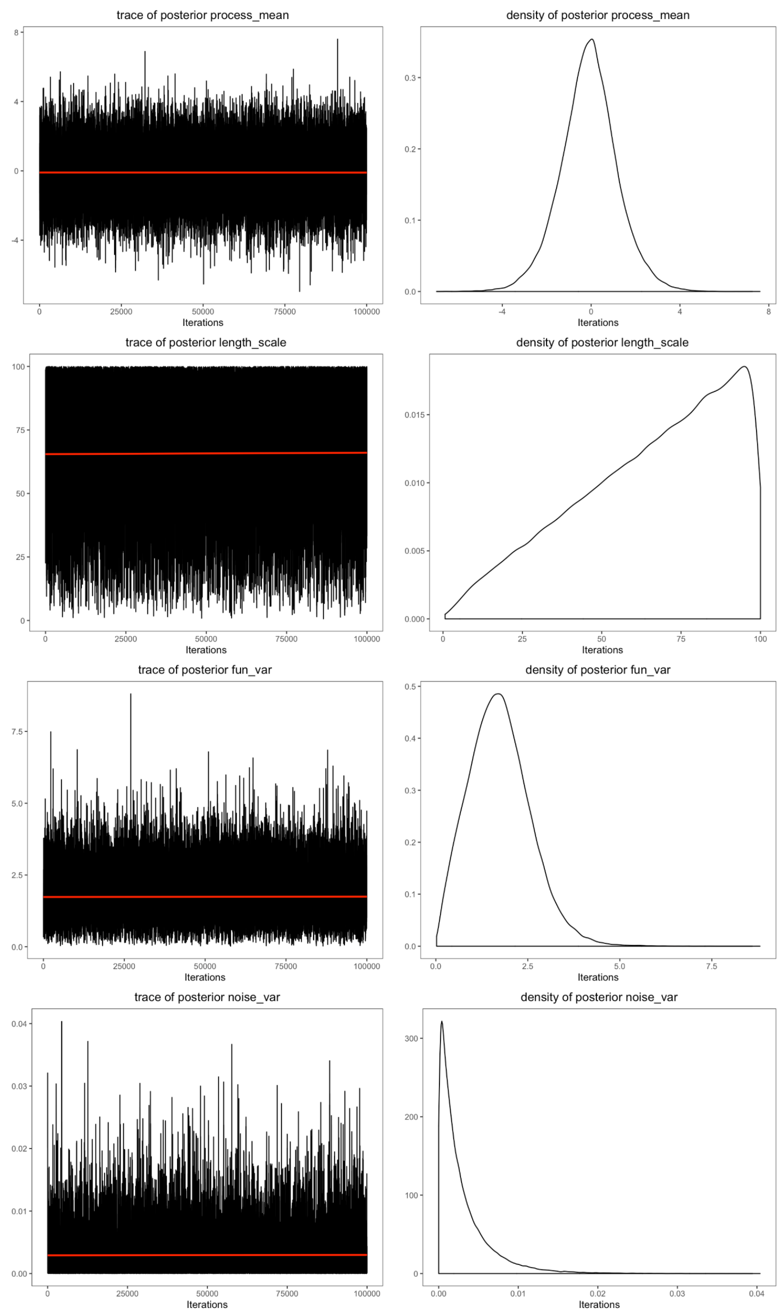

Figure 2. Trace plots and posterior densities generated by `BayesOptRegionGP`.

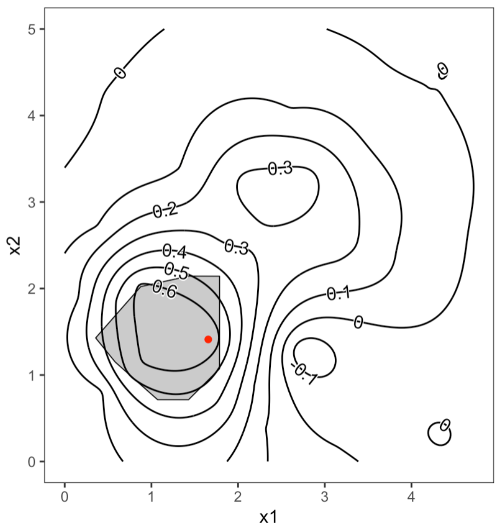

Figure 3. A credible region generated by `BayesOptRegionGP`.

```
  constr_lb = c(-sqrt(2), -sqrt(2)), constr_ub = c(sqrt(2), sqrt(2)),
  alpha = 0.05, B = 500
)
```

where `B` specifies the size of the bootstrap. The function returns a list. Use command `str(res)` to check its structure:

```
List of 6
 $ optima   :'data.frame': 475 obs. of  2 variables:
  ..$ X1: num [1:475] -0.206 -0.463 -0.144 -0.148 -0.297 ...
  ..$ X2: num [1:475] -0.2201 0.0449 -0.1123 -0.1345 -0.0745 ...
 $ opt_bag  :'data.frame': 1 obs. of  2 variables:
  ..$ X1: num -0.235
  ..$ X2: num -0.107
 $ design   :'data.frame': 22 obs. of  2 variables:
  ..$ X1: num [1:22] -1 1 -1 1 -1.41 ...
  ..$ X2: num [1:22] -1 -1 1 1 0 ...
 $ y        : num [1:22, 1] 87.6 86 87.3 83.3 86.9 ...
 $ constr_lb: num [1:2] -1.41 -1.41
 $ constr_ub: num [1:2] 1.41 1.41
 - attr(*, "class")= chr "crquad"
```

where `opt_bag` contains the bootstrap aggregated optimum. Use command:

```
plot(res, xlab = "x1", ylab = "x2")
```

to draw the confidence region on the optimum, as shown in Figure 4. The red dots indicates the bootstrap aggregated optimum.

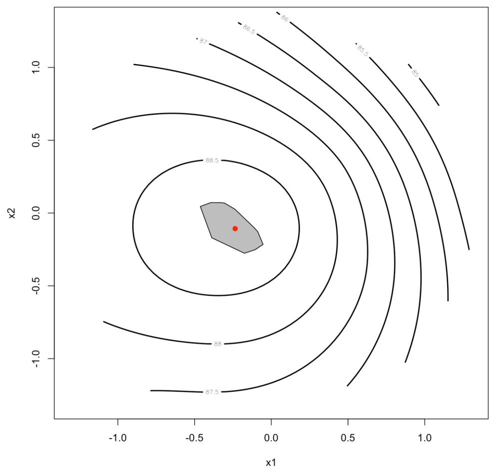

Figure 4. A confidence region generated by `OptRegionQuad`.

Function 4. `GloptiPolyRegion`

Function `GloptiPolyRegion` computes a distribution-free confidence region on the global optimum of a polynomial model up to cubic order in 3 $\sim$ 5 variables. To call it, use command:

```
res <- GloptiPolyRegion(
  design = dataset_4$design, y = dataset_4$y,
  constr_lb = rep(0, 5), constr_ub = rep(5, 5),
  alpha = 0.05, B = 500, degree = 3
)
```

where `degree` specifies the order of the polynomial model. The function returns a list. Use command `str(res)` to check its structure:

```
List of 4
 $ optima   :'data.frame': 475 obs. of  5 variables:
  ..$ X1: num [1:475] 5 2.07 5 5 5 ...
  ..$ X2: num [1:475] 2.6 2.37 2.39 2.26 2.39 ...
  ..$ X3: num [1:475] 0.714 1.13 0.992 0.852 1.219 ...
  ..$ X4: num [1:475] 2.84 2.66 2.6 2.61 2.54 ...
  ..$ X5: num [1:475] 2.64 2.63 2.42 2.21 2.37 ...
 $ opt_bag  :'data.frame': 1 obs. of  5 variables:
  ..$ X1: num 4.01
  ..$ X2: num 2.37
  ..$ X3: num 0.993
  ..$ X4: num 2.61
  ..$ X5: num 2.48
 $ constr_lb: num [1:5] 0 0 0 0 0
 $ constr_ub: num [1:5] 5 5 5 5 5
 - attr(*, "class")= chr "crpoly"
```

Use command:

```
plot(res, axes_labels = c("x1", "x2", "x3", "x4", "x5"))
```

to draw pairwise projections of the confidence confidence on the optimum, as shown in Figure 5

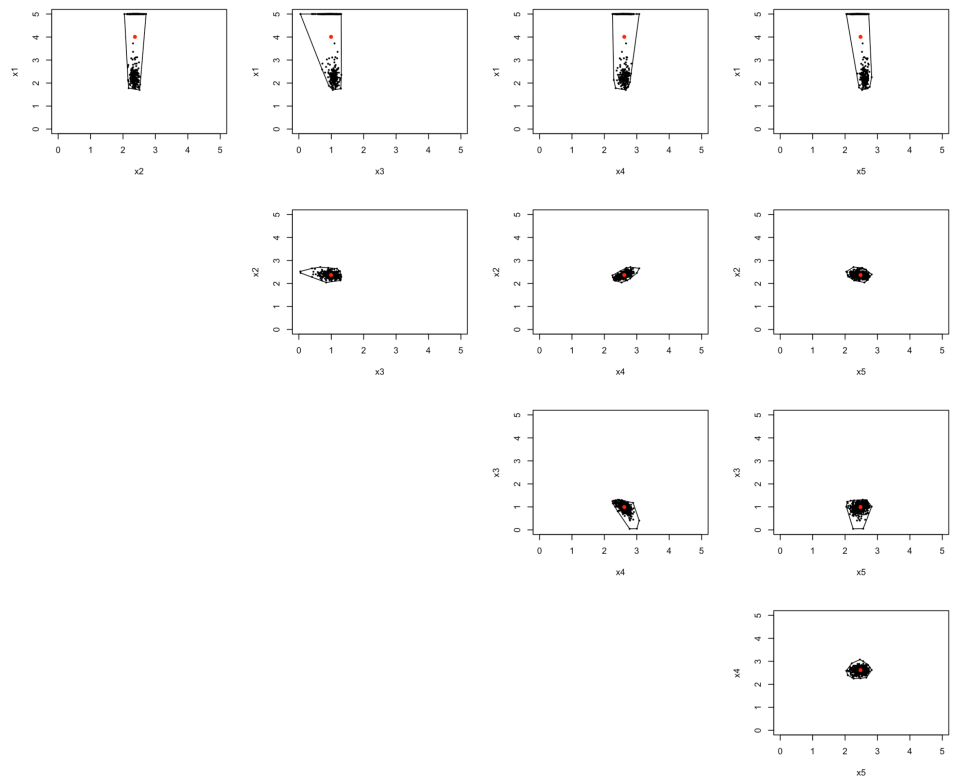

Figure 5. Projections of a confidence region generated by `GloptiPolyRegion`.

Function 5. `OptRegionTps`

Function `OptRegionTps` computes a distribution-free confidence region on the global optimum of a thin-plate spline model in 2 variables. To call it, use command:

```
res <- OptRegionTps(
  design = dataset_5$design, y = dataset_5$y,
  constr_lb = c(0, 0), constr_ub = c(5, 5),
  alpha = 0.05, B = 500
)
```

The function returns a list. Use command `str(res)` to check its structure:

```
List of 7
 $ optima   :'data.frame': 475 obs. of  2 variables:
```

```
  ..$ X1: num [1:475] 0.883 0.632 0.949 0.993 0.72 ...
  ..$ X2: num [1:475] 1.54 1.38 1.41 1.25 1.33 ...
 $ opt_bag  :'data.frame': 1 obs. of  2 variables:
  ..$ X1: num 0.757
  ..$ X2: num 1.26
 $ design   :'data.frame': 300 obs. of  2 variables:
  ..$ X1: num [1:300] 0.837 0.795 3.502 4.912 2.294 ...
  ..$ X2: num [1:300] 2.429 1.01 1.299 2.962 0.635 ...
 $ y        : num [1:300, 1] 0.36574 0.53458 -0.06362 0.00867 0.12086 ...
 $ lambda   : num 0.04
 $ constr_lb: num [1:2] 0 0
 $ constr_ub: num [1:2] 5 5
 - attr(*, "class")= chr "crtps"
```

where `lambda` is the penalization parameter value used to fit Thin Plate Spline model. Use commmand:

```
plot(res, xlab = "x1", ylab = "x2")
```

to draw the confidence region on the optimum, as shown in Figure 6.

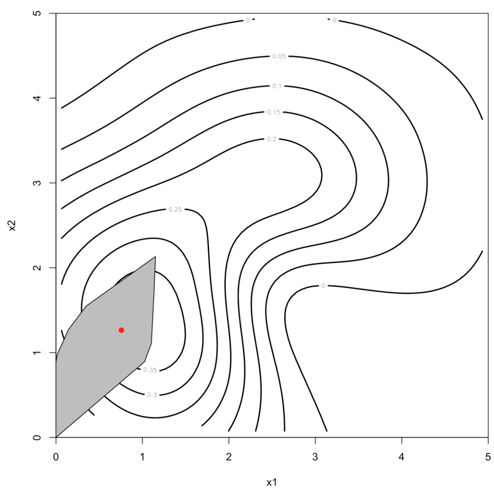

Figure 6. A confidence region generated by `OptRegionTps`.

Function 6. `OptRegionGP`

Function `OptRegionGP` computes a confidence region on the global optimum of a Gaussian process model in 2 variables. To call it, use command:

```
res <- OptRegionGP(
  design = dataset_6$design, y = dataset_6$y,
  constr_lb = c(0, 0), constr_ub = c(5, 5),
  alpha = 0.05, B = 1000, xi = 0.1, parallel = TRUE
)
```

The function returns a list. Use command `str(res)` to check its structure:

```
List of 5
 $ optima   :'data.frame': 950 obs. of  2 variables:
  ..$ x1: num [1:950] 1.07 1.07 1.07 1.07 1.07 ...
  ..$ x2: num [1:950] 1.43 1.43 1.43 1.43 1.43 ...
 $ opt_bag  :'data.frame': 1 obs. of  2 variables:
  ..$ x1: num 1.12
  ..$ x2: num 1.84
 $ rs_hat   :function (x)
  ..- attr(*, "srcref")= 'srcref' int [1:8] 146 3 152 3 3 3 146 152
  .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile'
          <environment: 0x7fcb0a634480>
 $ constr_lb: num [1:2] 0 0
 $ constr_ub: num [1:2] 5 5
 - attr(*, "class")= chr "crgpok"
```

Use commmand:

```
plot(res, xlab = "x1", ylab = "x2")
```

to draw the confidence region on the optimum, as shown in Figure 7.

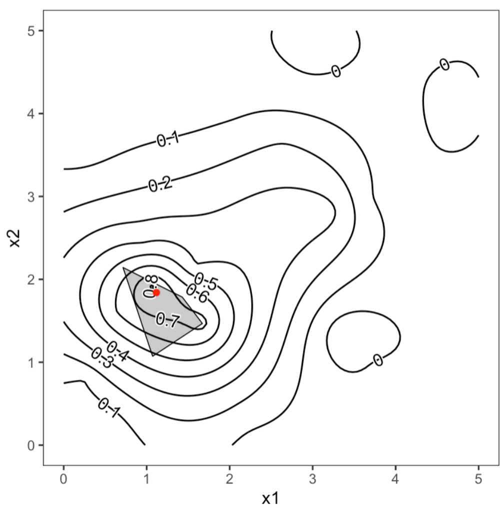

Figure 7. A confidence region generated by `OptRegionGP`.

Function 7. `OptRegionOK`

Function `OptRegionOK` computes a distribution-free confidence region on the global optimum of an ordinary Kriging model in 2 variables. To call it, use command:

```
res <- OptRegionOK(
  design = dataset_7$design, y = dataset_7$y,
  constr_lb = c(0, 0), constr_ub = c(5, 5),
```

```
  alpha = 0.05, B = 1000, xi = 0.1, parallel = TRUE
)
```

The function returns a list. Use command `str(res)` to check its structure:

```
List of 5
 $ optima   :'data.frame': 950 obs. of  2 variables:
  ..$ x1: num [1:950] 1.06 1.06 1.06 1.06 1.06 ...
  ..$ x2: num [1:950] 1.91 1.91 1.91 1.91 1.91 ...
 $ opt_bag  :'data.frame': 1 obs. of  2 variables:
  ..$ x1: num 1.13
  ..$ x2: num 1.78
 $ rs_hat   :function (x)
  ..- attr(*, "srcref")= 'srcref' int [1:8] 146 3 152 3 3 3 146 152
  .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile'
          <environment: 0x7fcb0a634480>
 $ constr_lb: num [1:2] 0 0
 $ constr_ub: num [1:2] 5 5
 - attr(*, "class")= chr "crgpok"
```

Use commmand:

```
plot(res, xlab = "x1", ylab = "x2")
```

to draw the confidence region on the optimum, as shown in Figure 8.

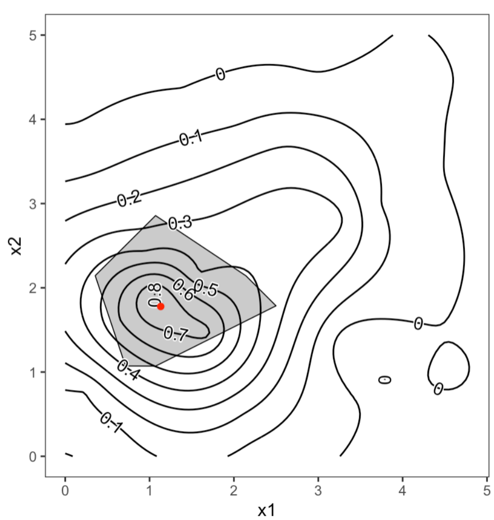

Figure 8. A confidence region generated by `OptRegionOK`.

Function 8. `OptRegionPC1Quad`

Function `OptRegionPC1Quad` takes high-dimensional observations in two controllable factors and compute a confidence region on the optimum of the assumed quadratic probabilistic first principal response surface. To call it, use command:

```
c(cr, ppca) %<-% OptRegionPC1Quad(
  design = dataset_8$design, Y = dataset_8$Y,
  constr_lb = c(-sqrt(2), -sqrt(2)), constr_ub = c(sqrt(2), sqrt(2)),
  alpha = 0.05, B = 500
)
```

where Y takes the observation matrix. The function returns two lists, `cr` and `ppca`. List `ppca` contains the information of the fitted PPCA model. Use command `str(ppca)` to check its structure:

```
List of 5
 $ mu_hat    : num [1:30] 4.84 -1.73 -2.52 1.88 2.54 ...
 $ sigma2_hat: num 86.3
 $ W_hat     : num [1:30, 1] 8.329 -0.0943 0.4814 3.9147 0.5941 ...
 $ z_given_x : num [1, 1:22] -0.1061 -0.8549 -0.0923 0.0157 -0.9758 ...
 $ props     : num [1:30] 0.1451 0.1263 0.1039 0.0886 0.0878 ...
```

where `mu_hat` contains the fitted high-dimensional mean vector, `sigma2_hat` contains the fitted noise variance , `W_hat` contains the fitted dimension-transformation matrix, `z_given_x` contains the probabilistic first principal responses at the design points, and `props` contains the standard PCA component variance proportions. List `cr` contains the information of the confidence region. Use command `str(cr)` to check its structure:

```
List of 6
 $ optima   :'data.frame': 475 obs. of  2 variables:
  ..$ X1: num [1:475] 0.0997 0.0314 -0.1865 -0.0868 -0.0783 ...
  ..$ X2: num [1:475] -0.1826 -0.0818 -0.1719 -0.0881 -0.0716 ...
 $ opt_bag  :'data.frame': 1 obs. of  2 variables:
  ..$ X1: num -0.0424
  ..$ X2: num -0.146
 $ design   :'data.frame': 22 obs. of  2 variables:
  ..$ X1: num [1:22] -1 1 -1 1 -1.41 ...
  ..$ X2: num [1:22] -1 -1 1 1 0 ...
 $ y        : num [1:22] -0.1061 -0.8549 -0.0923 0.0157 -0.9758 ...
 $ constr_lb: num [1:2] -1.41 -1.41
 $ constr_ub: num [1:2] 1.41 1.41
 - attr(*, "class")= chr "crquad"
```

Use command:

```
plot(cr, xlab = "x1", ylab = "x2")
```

to draw the confidence region on the optimum, as shown in Figure 9.

Function 9. `GloptipolyR`

Function `GloptiPolyR` implements the Gloptipoly (**?**) algorithm. Consider optimizing the following quadratic function in three variables:

$$f(\boldsymbol{x}) = -1.5x_1 + 2.13x_2 - 1.81x_3 + 7.13x_1x_2 + 3.27x_1x_3 + 2.73x_2x_3 + 4.69x_1^2 + 6.27x_2^2 + 5.21x_3^2$$

defined over $\mathcal{X} = \{-2 \leq x_i \leq 2, i = 1, 2, 3\}$, with its global minimum at $\boldsymbol{x}^* = (0.46,$ -0.46, 0.15). The optimization problem can be formally written as:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{subject to:} \quad & g_1(\boldsymbol{x}) = x_1 + 2 \geq 0 \end{aligned}$$

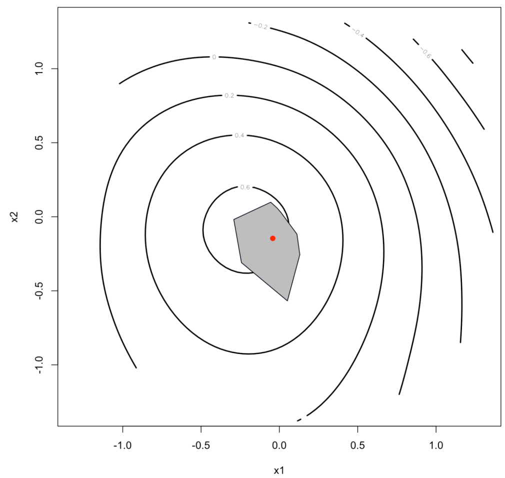

Figure 9. A confidence region generated by `OptRegionPC1Quad`.

$$\begin{aligned} g_2(\boldsymbol{x}) &= x_1 - 2 \le 0 \\ g_3(\boldsymbol{x}) &= x_2 + 2 \ge 0 \\ g_4(\boldsymbol{x}) &= x_2 - 2 \le 0 \\ g_5(\boldsymbol{x}) &= x_3 + 2 \ge 0 \\ g_6(\boldsymbol{x}) &= x_3 - 2 \le 0 \end{aligned}$$

The input for `GloptiPolyR` is a list `P` of seven sub-lists, corresponding to $f(\boldsymbol{x})$, $g_1(\boldsymbol{x})$, $g_2(\boldsymbol{x})$, $\cdots$, $g_6(\boldsymbol{x})$, respectively:

```
P <- list()
p_f <- list()
p_g_1 <- list(); p_g_2 <- list(); p_g_3 <- list()
p_g_4 <- list(); p_g_5 <- list(); p_g_6 <- list()
```

Each of these seven sub-lists has two elements: (1) a multi-dimensional array, denoted by 'c', and (2) an attribute, denoted by 't'. The multi-dimensional array is generated from the monomial coefficients of the corresponding polynomial function. The rule is to put the coefficient of the $x_1^i x_2^j x_3^k$ term in the $[i+1, j+1, k+1]$ position of the array, and place zeroes in other positions:

```
p_f$c <- array(0, dim = c(3, 3, 3))
p_f$c[2, 1, 1] <- -1.5; p_f$c[1, 2, 1] <- 2.13; p_f$c[1, 1, 2] <- -1.81
p_f$c[2, 2, 1] <- 7.13; p_f$c[2, 1, 2] <- 3.27; p_f$c[1, 2, 2] <- 2.73
p_f$c[3, 1, 1] <- 4.69; p_f$c[1, 3, 1] <- 6.27; p_f$c[1, 1, 3] <- 5.21

p_g_1$c <- array(0, dim = c(3, 3, 3))
p_g_1$c[1, 1, 1] <- 2; p_g_1$c[2, 1, 1] <- 1

p_g_2$c <- array(0, dim = c(3, 3, 3))
p_g_2$c[1, 1, 1] <- -2; p_g_2$c[2, 1, 1] <- 1
```

```
p_g_3$c <- array(0, dim = c(3, 3, 3))
p_g_3$c[1, 1, 1] <- 2; p_g_3$c[1, 2, 1] <- 1

p_g_4$c <- array(0, dim = c(3, 3, 3))
p_g_4$c[1, 1, 1] <- -2; p_g_4$c[1, 2, 1] <- 1

p_g_5$c <- array(0, dim = c(3, 3, 3))
p_g_5$c[1, 1, 1] <- 2; p_g_5$c[1, 1, 2] <- 1

p_g_6$c <- array(0, dim = c(3, 3, 3))
p_g_6$c[1, 1, 1] <- -2; p_g_6$c[1, 1, 2] <- 1
```

Next, set the attribute for the objective function as either “min” or “max”:

```
p_f$t <- "min"
```

Then, set the attributes for the constraint functions as either “¿=” or “¡=”:

```
p_g_1$t <- ">="; p_g_2$t <- "<="
p_g_3$t <- ">="; p_g_4$t <- "<="
p_g_5$t <- ">="; p_g_6$t <- "<="
```

Finally, we construct the list `P` from the 7 sub-lists and use it to call `GloptiPolyR`:

```
P <- list(p_f, p_g_1, p_g_2, p_g_3, p_g_4, p_g_5, p_g_6)
res <- GloptiPolyR(P)
```

then, use command `str(res)` to retrieve the optimization result:

```
List of 2
 $ solution : num [1:3] 0.46 -0.465 0.151
 $ objective: num -0.977
```